An implementation of graph based SLAM with unknown correspondence

Yiming Hu
March 8, 2021

1 INTRODUCTION

In this report, we present a traditional graph based SLAM algorithm on a well-recognized online SLAM dataset called Victoria Park.

Graph SLAM is an interesting topic that extends from this course. It treats information as nodes and edges and optimizes a topological graph based on constrains. It drops the Markov assumption and performs as a full SLAM algorithm. Here, it is implemented by us using pure Matlab code without any optimization tools.

We cover the most interesting and valuable exploration in the method part. For example, we applied a preliminary data association algorithm using maximum likelihood to deal with unknown correspondence. And in order to reduce space complexity and build a more solid topological graph, we raised an idea to merge consecutive robot poses where happens no observation.

In our experiment part, we present the optimized result on a small part of the dataset as well as a complete graph optimization using an online package. A detail analysis in this part further explains the limitation of Graph SLAM and a few of my thinking into it.

The structure of this report is arranged as follows. Part 2 covers the background of Simultaneous localization and mapping(SLAM). Due to page limitation, we shall present information related to Victoria Park Dataset **here**. Part 3 explains three most interesting methods we used

in our algorithm. Part 4 shows the result of our simulation with a few comments and analysis. Part 5 summarizes the work we have done as well as feelings throughout .

2 BACKGROUND

To search and navigate, one intelligent agent needs first to locate itself, which can be easily done with Global Positioning System (GPS). However, GPS signal is usually unreliable in indoor scenario, when it is a perfect timing for robot to explore, build and update its own local map using simultaneous localization and mapping (SLAM). To perform SLAM, robot needs to collect several kinds of data. Odometry data u_t is usually used to initialize the robot state and the observation data z_t, with correspondence c_t, provides additional information to acquire an optimal state prediction $\hat{x}_t$. Landmarks y_t are global marks to help robot find its position. A simple illustration is shown in Figure 2.1.

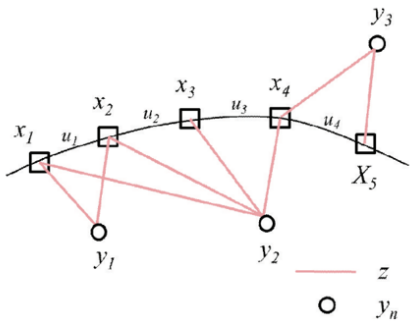

Figure 2.1: Links between state, control and landmark

Sometimes, there even exists other bunches of data from IMU to be fused. Therefore, an series of advanced filtering algorithm has been applied to achieve a balanced belief and guarantee the most reliable state.
Rudolf E. Kálmán proposes a filter algorithm called Kalman Filter to deal with linear optimization. EKF SLAM using extended Kalman filter to deal with nonlinear motion model based on the model:

$$p(x_t, m|z_{1:t}, u_{1:t}, x_0) = p(z_t|x_t, m) \int p(x_t|u_t, x_{t-1}) p(x_{t-1}) d_{x_{t-1}}$$

By loosing Gaussian assumption to Markov assumption, particle filters or Monte Carlo (MC) method gain a popularity especially for its ability to solve kidnap problems. it plays an important role in Fast SLAM and expresses posterior distribution $p(x_t^{[k]})$ of each particle k by:

$$p(x_{1:t}^{[k]}|z_{1:t}, u_{1:t}, c_{1:t-1}) = p(x_t^{[k]}|x_{t-1}^{[k]}, u_t) p(x_{1:t-1}^{[k]}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$$

It is worth noticing that EKF SLAM and Fast SLAM are both online SLAM since they will drop previous data once finishing updating based on Markov assumption. Graph SLAM, instead,

incorporates all the data and deal with an error minimization problem. The goal is to find the optimal state x^* that maximize the likelihood of observations $F(x)$ [2]:

$$x = \arg\max_{x} \mathrm{F}(x)$$
$$F(x) = \Sigma e_{ij}^{T} \Omega_{ij} e_{ij}$$

While the error e_{ij} depends on the difference between the real and expected measurement between node i and j:

$$e_{ij} = z_{ij} - \hat{z_{ij}}(n_i, n_j)$$

The main idea of Graph SLAM is to solve a graph optimization problem[2]. Nodes in the graph are typically robot poses$x_{1:t}$ and map features m_j. Edges linking nodes correspond to an event of observation or motion. A sparse $n \times n$ information matrix Ω is maintained to achieve optimal solution, while n equals to total number of nodes n.

Method	Pros	Cons
EKF	- Able to deal with non-linear model. - Can provide good loop closure reference. - Large body of work.	- Problematic when dealing with large maps that lead to numerous issues.
FastSLAM	- Can accommodate any distribution. - Improve computational speed. - Require less storage memory. - Better data association accuracy	- Sampling data degeneracy issues. - Particle depletion during sampling process.
Graph SLAM	- Can provide better accuracy and consistency in its estimating technique. - Able to consume and process large area of mapping.	- Very high computational cost.

Figure 2.2: A comparison between popular SLAM algorithm

A traditional Graph SLAM algorithm includes several steps shown in the table 2.1. These steps also are the foundation of our algorithm.
A simple comparison between EKF SLAM, Fast SLAM and Graph SLAM algorithm is presented in Figure 2.2 [7]
However, due to high time consumption for real time, graph based SLAM was not commonly used until graph optimizer like g2o[4],cere-solver, or iSAM[3] comes out. They develop advanced methods to deal with linear sparse system based on standard methods like Levenberg-Marquardt (LM). It ensures that Graph SLAM is fast enough to be implemented for realtime scenarios. Graph SLAM can deal with dynamic landmarks with mobility detection[6]. It can be

	Input/Output	Main idea
Step 1: Initialization	$u_{1:t}, x_0, \mu_{1:t}$	Find a suitable mean for linearization
Step 2: Linearization	$\mu_{1:t}, z_{1:t}, u_{1:t}/\Omega, \xi$	Integrate "constrains" of motion and measurement
Step 3: Reduce	$\Omega, \xi/\bar{\Omega}, \bar{\xi}$	Remove features and add new constrains to states
Step 4: Solve	$\Omega, \xi, \bar{\Omega}, \bar{\xi}/\bar{\mu}_{1:t}$	Compute new means and covariance

Table 2.1: Graph SLAM algorithm[8]

applied even to 3D space for real-time exploration of the unknown environment by Unmanned Aerial Vehicles(UAV)[1].

3 Method

3.1 Landmark with signature

Woods are perfect landmarks in Victoria Park since they are static and easy to detect. The radius of the trunk for each wood should be an unique value that serves as their own signature for data association. Although the radius is initially unknown, fortunately, it can be deducted by observation data.

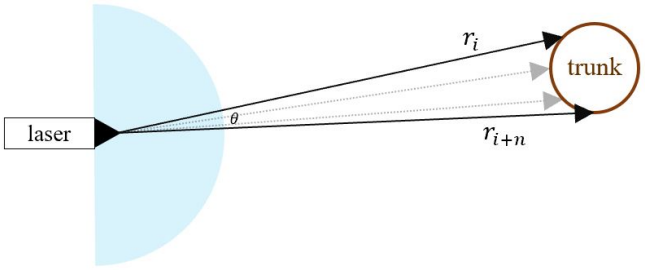

Figure 3.1: Measurement received after detecting woods

The geometric centre of a landmark m_j is measured by range and bearing observation and its radius/signature:

$$\begin{bmatrix} m_{j,x} \\ m_{j,y} \end{bmatrix} = \begin{bmatrix} (r_t^i + m_{j,s})\cos(\phi_t^i + \theta_t - \pi/2) \\ (r_t^i + m_{j,s})\sin(\phi_t^i + \theta_t - \pi/2) \end{bmatrix}$$

While the signature is related to the number of bundles as well as the range, assuming $\Delta\theta$ is a very small number.:

$$\Delta\theta = \frac{n \cdot \pi}{N},\ N = 360$$
$$s = \frac{\Delta\theta}{2} \cdot \max(r_i, r_{i+n})$$

Then naturally rises an interesting question: what is the reasonable observation error σ_s^2 for signature? Intuitively, if a tree is far away or has a thinner trunk, it is harder to acquire precise measurement. The largest error for $\Delta\theta$ is $2\pi/N$ (imagining two bundle of laser pass through the edge of a wood). This leads to a signature error of $0.3m$ with range measurement of 30 meters. In practice, we set observed signature error τ_s^2 to 0.1.

3.2 MERGING MOTION STATE

In our algorithm, several consecutive motion without observation data is merged to decrease the dimension of information matrix Ω. Assuming we are going to merge $x_i, x_{i+1}...x_{i+n}$, to get merged G_t and u_t. The key lies on the starting point x and end point x'. According to motion model:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} v_t\Delta_t(\cos\theta - \frac{\tan\alpha}{L}(a\sin\theta + b\cos\theta)) \\ v_t\Delta_t(\sin\theta + \frac{\tan\alpha}{L}(a\cos\theta - b\sin\theta)) \\ v_t\Delta_t\frac{\tan\alpha}{L} \end{bmatrix}$$

Hence,

$$v_t^2\Delta_t^2(1 + \frac{\tan\alpha^2}{L^2}(a^2+b^2) - \frac{2\tan\alpha}{L}b) = (x'-x)^2 + (y'-y)^2 = \Delta_d^2$$
$$v_t\Delta_t\frac{\tan\alpha}{L} = \theta' - \theta = \Delta_\theta^2$$

From which we get merged control data:

$$v_t = \frac{b\Delta_\theta + \sqrt{\Delta_d^2 - a^2\Delta_\theta^2}}{t}$$
$$\tan\alpha = \frac{L}{b + \sqrt{\frac{\Delta_d^2}{\Delta_\theta^2} - a^2}}$$

The purpose of merging is to : ① reduce the space complexity as the original dataset has weigh more motion states than observations ② release redundant constrains and speed up the optimization.

The optimization system could be seen as a spring and mass system. Each edge, generated by either motion or measurement, adds rigid constrains on its neighbours. We define a node with observation as a tightly constrained node and a node without observation as a loosely constrained node. We'd like to reduce the latter kind of node as they contribute less to the optimization and have more uncertainty. However, the main reason to do this is to decrease space complexity since only one in around seven states has measurement data.

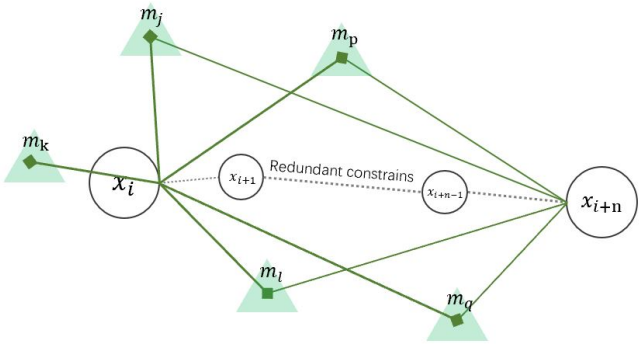

Figure 3.2: Edge constrains between motion states and landmarks

3.3 Data association and Outlier detection

In our implementation, we assume unknown landmark correspondence and thus data association becomes a big problem. Our idea is to add each observation as a new landmark in the map since graph based SLAM is a full SLAM and then resort to a practical method to merge landmarks.

To merge a tremendous number of landmarks to a few clustered representatives, we think of two methods. One is K-means clustering and the other is maximum likelihood test. K-means clustering is faster since it only makes comparison between landmarks and cluster means. However, the effect of K-meams clustering depends largely on its initialization. So we decide to use maximum likelihood test to merge landmarks.

The probability distribution $p(x,m)$ over state x and landmark m is expressed as the following form:

$$\Omega = \begin{pmatrix} \Omega_{xx} & \Omega_{xm} \\ \Omega_{mx} & \Omega_{mm} \end{pmatrix}, \ \xi = \begin{pmatrix} \xi_x \\ \xi_x \end{pmatrix}$$

The marginal probability $p(m)$ is represented as :

$$\bar{\Omega}_{mm} = \Omega_{mm} - \Omega_{mx}\,\Omega_{xx}^{-1}\,\Omega_{xm}$$

The probability of landmark m_j and m_k being the same landmark is proportion to the following equation:

$$p \simeq \mid 2\pi\Omega_{\Delta_{jk}}^{-1} \mid^{-\frac{1}{2}} exp\{-\frac{1}{2}\mu_{\Delta_{jk}}^{T}\Omega_{\Delta_{jk}}^{-1}\mu_{\Delta_{jk}}\}$$

Here we shall define our data association alogrithm as:

The computational complexity of one complete data association is $O(N^2)$ and N is the number of landmarks in the map. It should be noticed that it is possible to reduce it to linear complexity $O(kN)$ using other association method like K-means. Therefore, it exists large space for improvement in this part.

An outlier detection is implemented right after data association. This simply by checking the number of effective observations to a single landmark $n_{\tau_j} = size(\tau_j)$. A landmark will be classified as an temporal outlier if $n_{\tau_j} < threshold$ which we set to be around 3.

Algorithm 1: correspondence test

Result: new correspondence c
for *each landmark j* **do**
  for *each landmark k afterwards* **do**
    Caculate p = $\simeq |\, 2\pi\Omega^{-1}_{\Delta_{jk}} \,|^{-\frac{1}{2}} \; exp\{-\frac{1}{2}\mu^{T}_{\Delta_{jk}}\Omega^{-1}_{\Delta_{jk}}\mu_{\Delta_{jk}}\}$
    if *p>threshold* **then**
      set $c_k = c_j$
    end
  end
end

4 Experiment

In experiment part, we run a simulation on Victoria park dataset both using our algorithm and an online ***Matlab-Graph-Optimization*** package provided by Roma Teng. It is stated here that our algorithm is able to handle light state graph but not for such a large dataset like Victoria park due to unknown correspondence. Even for **Matlab-Graph-Optimization**, they choose to optimize a pre-processed graph with accurate correspondence. It is perfectly in line with the trade of dividing SLAM into front-end and back-end.

4.1 A light trial on Victoria park

To test our own algorithm, we run several trials on an initial part of Victoria park dataset and compare our result to ground truth. As shown in figure 5.1, GPS data is not always available and may have some tiny disturbance. Nevertheless, it can be seen our optimized path is closer to the ground truth than the odometry path.

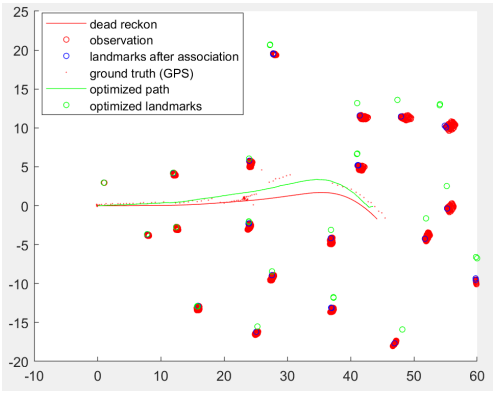

Figure 4.1: Victoria park trial with 2000 states

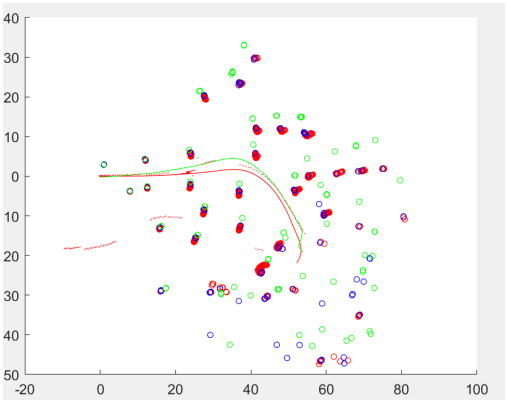

Figure 4.2: Victoria park trial with 2500 states

Our landmark correspondence result(points in blue in figure 5.3) seems to be successful in the light trial as it clusters similar cluster to one representative. It further adds effective constrains

to nodes and ensure a reliable optimization.

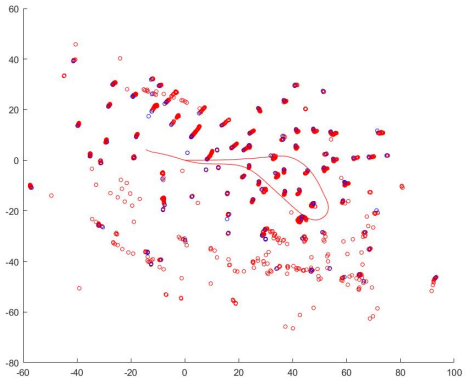

Figure 4.3: Landmark correspondence and outlier detection

Outliers take place due to two cases : ① wrong observations (should be eliminated forever) ② large deviation of location(should be abandoned temporarily and included later on). They are excluded in the optimization process as wrong correspondence will seriously undermine the result.
However, the landmark correspondence algorithm may not work as smoothly when applied to a full dataset. This is mainly because as position error accumulates with time, it becomes harder to recognize the same landmark because of the relative rotate and tranlation of the map. In our scenario, since we resort to a likelihood test based on threshold, it may even lead to wrong correspondences, as illustrated in figure 4.4. This is disastrous for graph SLAM because a wrong correspondence will add large disturbance to the whole system as it runs a full-scale optimization minimizing $F(x) = \Sigma e_{ij}^T \Omega_{ij} e_{ij}$.

Figure 4.4: Landmark drifting and wrong association

As a result, for state-of-art SLAM algorithm, it is wise and well-recognized to split Graph SLAM into front-end and back-end. The responsibility to construct initial map and find correct data association lies on the front-end so that back-end can concentrate on optimizing the graph. A

feasible stragety for data association in the front end is to run an EKF SLAM and find a roughly correct correspondence.

4.2 A FULL SIMULATION USING G2O LIBRARY

In this part, we resort to a online library to present the effect of Graph SLAM. which you can find here(https://github.com/UTS-CAS/Matlab-Graph-Optimization.html). The total running time is 1336 seconds, which is surprisingly close to the cost of FastSLAM[5]. As shown in figure 4.5, the optimized graph accords with ground truth at first but has a increasing deviation later. This is an interesting phenomenon happening in our algorithm as well.

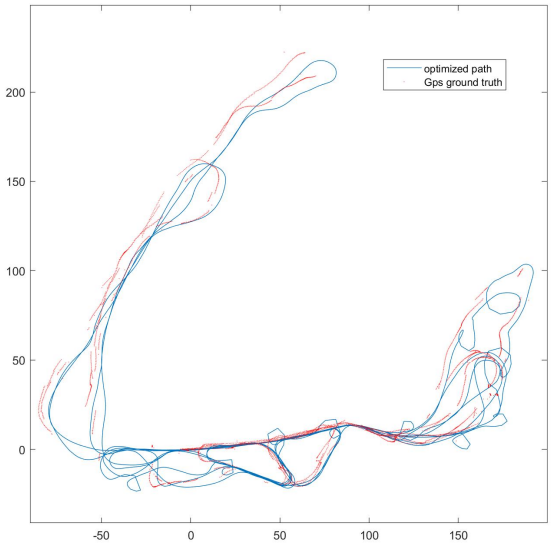

Figure 4.5: Landmark drifting and wrong association

As shown in figure 4.6, the cost manages to converge to a small value after around 8 iterations. Therefore, the problem may not lie on the optimization algorithm.
The error, in our perspective, is mainly due to measurement error as well as data drifting during data collection. In fact, the size of trunk may not be a standard circle. The woods sometimes will block each other as well.Besides, the fact that observation of nearer landmarks is more accurate than farther ones was not taken into account in our implementation.
Data drifting is also a fatal problem for Graph SLAM which sometimes happens during a long-time dataset collection. Imagine we have two measurement z_{j,t_1}, z_{k,t_1} at time t_1, pose x_{t_1} the error function is expressed as:

$$F(x) = \bar{F}(x) + e_j^T \Omega_{t_1} e_j + e_k^T \Omega_{t_1} e_k$$

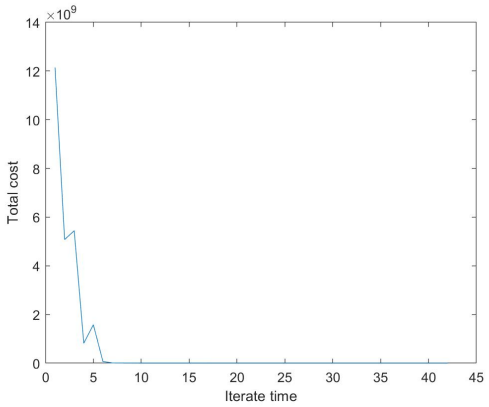

Figure 4.6: Cost minimizing in graph optimization

Ideally, we are glad to decrease this appended error to zero with a optimal $\bar{x}_{t_1}$:

$$\arg\max_{\bar{x}_{t_1}} \mathrm{F}(x) = h^{-1}(z_{j,t_1}) = h^{-1}(z_{k,t_2})$$

However, if one measurement is slightly drifted, in order to minimize the joint cost, Graph SLAM algorithm will balance the pose. Thus, it shall be stated here a precise, stable, continous measurement during data collection is crucial for Graph SLAM.
Besides, an interesting idea, which I think will be able to improve optimization accuracy, is to split the graph segment and optimize them separately. The advantage is that we can tune the noise model as iteration parameters to fit each scenario. Finally, a point cloud matching algorithm is used to connect all the segment. Maybe I can try this in the future study.

5 Conclusion and acknowledgement

In this project, we have done a systematic learning and implementation of Graph SLAM algorithm on Victoria park dataset and presented simulation results as well as our analysis.

The main contribution of our project is to bring the idea in practice to reduce the spatial and computational dimension of graph optimization problem. They correspond to decreasing the number of nodes and edges respectively. In our project, we tried to merge robot poses and decreased the dimension of states around six to seven times. Another idea, is to take a sparse subset of observation for higher computational speed as well as, to split the graph into segment for higher accuracy. This deserves our further research in future's study.

The gain of this project is weigh more than its contribution. We made an brave attempt to both front-end and back-end and tried various methods to deal with raw data and optimize our algorithm. We kept continuous discussion and analysis in order to find solution to the question popping up one by one. I realized that there is still a long way to go until I reach the

frontier of SLAM research. And this project, ignites my passion to keep such study.

I shall here send my best respect to my teammate, Jiarui Tan and to this project, this course. Thank you letting me have such a chance to challenge myself.

REFERENCES

[1] A. Annaiyan, M. A. Olivares-Mendez, and H. Voos. Real-time graph-based slam in unknown environments using a small uav. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1118–1123, 2017.

[2] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 12 2010.

[3] M. Kaess, A. Ranganathan, and F. Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[4] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. pages 3607 – 3613, 06 2011.

[5] Xu Lei, Bin Feng, Guiping Wang, Weiyu Liu, and Yalin Yang. A novel fastslam framework based on 2d lidar for autonomous mobile robot. *Electronics*, 9:695, 04 2020.

[6] Lingzhu Xiang, Zhile Ren, Mengrui Ni, and O. C. Jenkins. Robust graph slam in dynamic environments with moving landmarks. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2543–2549, 2015.

[7] Talha Takleh Omar Takleh, Nordin abu bakar, Shuzlina Rahman, Raseeda Hamzah, and Zalilah Abd Aziz. A brief survey on slam methods in autonomous vehicle. *International Journal of Engineering and Technology(UAE)*, 7:38–43, 11 2018.

[8] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.